AED2 - Algoritmos e Estr. de Dados II

Aula 14: Introdução a Grafos

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

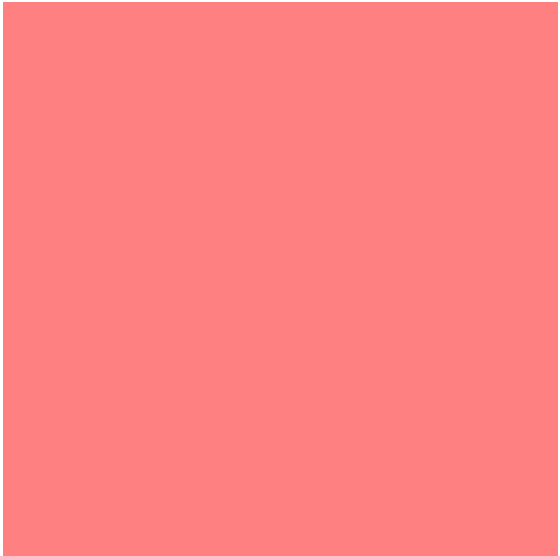
Fevereiro de 2026

Sumário

1. O Mundo Conectado

Um **Grafo** $G(V, E)$: **V**értices e **A**restas.
Grafos modelam **relacionamentos**:

- **Redes Sociais:** Vértices = Pessoas, Arestas = Amizades.
- **Mapas:** Vértices = Cidades, Arestas = Estradas.
- **Web:** Vértices = Páginas, Arestas = Hyperlinks.



2. Como representar no computador?

A escolha da representação afeta drasticamente a complexidade de tempo e memória. Seja V o número de vértices e E o número de arestas.

2.1. Matriz de Adjacência

Uma matriz M de tamanho $V \times V$, onde:

- $M[i][j] = 1$ (ou peso) se existe aresta de i para j.
- $M[i][j] = 0$ (ou ∞) se não existe.

Vantagens	**Desvantagens**
Aresta $u \to v$ em $O(1)$.	Memória $O(V^2)$.
Simples.	Vizinhos $O(V)$.

Uso: Apenas para grafos pequenos ($V \leq 2000$) ou muito densos ($E \approx V^2$).

2.2. Lista de Adjacência (Padrão Ouro)

Um vetor de listas. Para cada vértice u, armazenamos uma lista com seus vizinhos.

Vantagens	**Desvantagens**
Memória $O(V + E)$	Aresta (u, v) em $O(\text{grau})$

Uso: Padrão para 99

3. Implementação em Java

Em C++, usamos `vector<int> adj[]`. Em Java, temos um pequeno problema: arrays de genéricos (`ArrayList<Integer>[]`) causam warnings. Soluções comuns: 1. `ArrayList<ArrayList<Integer>>` (Mais OO, levemente mais lento e verboso). 2. `List<Integer>[]` com casting (Mais compacto).

Modelo Padrão (Adjacency List)

```
import java.util.*;

public class Grafo {
    // Vértices rotulados de 0 a V-1
    private int V;
    private List<List<Integer>> adj;

    public Grafo(int V) {
        this.V = V;
        adj = new ArrayList<>(V);
```

4. Edge List (Lista de Arestas)

Às vezes (ex: Kruskal para MST, Bellman-Ford), não precisamos saber quem são os vizinhos de u. Só precisamos da lista de todas as arestas.

```
class Edge implements Comparable<Edge> {
    int u, v, weight;
    // construtor...
    public int compareTo(Edge other) {
        return this.weight - other.weight;
    }
}

List<Edge> arestas = new ArrayList<>();
```

Isso simplifica a ordenação de arestas por peso.

5. Dicas de Entrada (Input)

Em problemas de juiz online, a entrada geralmente é dada assim:

```
5 6  (5 vértices, 6 arestas)
0 1
0 2
1 3
...
```

Template de Leitura:

```java
Scanner sc = new Scanner(System.in);
int V = sc.nextInt();
int E = sc.nextInt();

// Inicializa List<List<Integer>>...

for(int i=0; i<E; i++) {
    int u = sc.nextInt(); // Se vértices forem 1-based, use (u-1)
    int v = sc.nextInt();
```